

CSci 127: Introduction to Computer Science



hunter.cuny.edu/csci

Frequently Asked Questions

From lecture slips & recitation sections.

Frequently Asked Questions

From lecture slips & recitation sections.

- When is recitation? It's not on my schedule.

Frequently Asked Questions

From lecture slips & recitation sections.

- When is recitation? It's not on my schedule.
The course is hybrid:

Frequently Asked Questions

From lecture slips & recitation sections.

- When is recitation? It's not on my schedule.

The course is hybrid: 1.5 hours of lecture (T 9:45-11, 118 HN);

Frequently Asked Questions

From lecture slips & recitation sections.

- When is recitation? It's not on my schedule.

The course is hybrid: 1.5 hours of lecture (T 9:45-11, 118 HN); 0.5 hours quizzes (self-scheduled, 1001E HN);

Frequently Asked Questions

From lecture slips & recitation sections.

- When is recitation? It's not on my schedule.

The course is hybrid: 1.5 hours of lecture (T 9:45-11, 118 HN); 0.5 hours quizzes (self-scheduled, 1001E HN); 1.0 hours on-line lab exercises (do anywhere).

Frequently Asked Questions

From lecture slips & recitation sections.

- When is recitation? It's not on my schedule.
The course is hybrid: 1.5 hours of lecture (T 9:45-11, 118 HN); 0.5 hours quizzes (self-scheduled, 1001E HN); 1.0 hours on-line lab exercises (do anywhere).
- When is the midterm?

Frequently Asked Questions

From lecture slips & recitation sections.

- When is recitation? It's not on my schedule.
The course is hybrid: 1.5 hours of lecture (T 9:45-11, 118 HN); 0.5 hours quizzes (self-scheduled, 1001E HN); 1.0 hours on-line lab exercises (do anywhere).
- When is the midterm?
There is no midterm. Instead there's required weekly quizzes.

Frequently Asked Questions

From lecture slips & recitation sections.

- When is recitation? It's not on my schedule.
The course is hybrid: 1.5 hours of lecture (T 9:45-11, 118 HN); 0.5 hours quizzes (self-scheduled, 1001E HN); 1.0 hours on-line lab exercises (do anywhere).
- When is the midterm?
There is no midterm. Instead there's required weekly quizzes.
- When is the final?

Frequently Asked Questions

From lecture slips & recitation sections.

- When is recitation? It's not on my schedule.
The course is hybrid: 1.5 hours of lecture (T 9:45-11, 118 HN); 0.5 hours quizzes (self-scheduled, 1001E HN); 1.0 hours on-line lab exercises (do anywhere).
- When is the midterm?
There is no midterm. Instead there's required weekly quizzes.
- When is the final?
Monday, 16 December, 9-11am.

Frequently Asked Questions

From lecture slips & recitation sections.

- When is recitation? It's not on my schedule.
The course is hybrid: 1.5 hours of lecture (T 9:45-11, 118 HN); 0.5 hours quizzes (self-scheduled, 1001E HN); 1.0 hours on-line lab exercises (do anywhere).
- When is the midterm?
There is no midterm. Instead there's required weekly quizzes.
- When is the final?
Monday, 16 December, 9-11am.
- Can I submit late homework?

Frequently Asked Questions

From lecture slips & recitation sections.

- When is recitation? It's not on my schedule.
The course is hybrid: 1.5 hours of lecture (T 9:45-11, 118 HN); 0.5 hours quizzes (self-scheduled, 1001E HN); 1.0 hours on-line lab exercises (do anywhere).
- When is the midterm?
There is no midterm. Instead there's required weekly quizzes.
- When is the final?
Monday, 16 December, 9-11am.
- Can I submit late homework?
No. Instead we drop the 5 lowest grades.

Frequently Asked Questions

From lecture slips & recitation sections.

- When is recitation? It's not on my schedule.
The course is hybrid: 1.5 hours of lecture (T 9:45-11, 118 HN); 0.5 hours quizzes (self-scheduled, 1001E HN); 1.0 hours on-line lab exercises (do anywhere).
- When is the midterm?
There is no midterm. Instead there's required weekly quizzes.
- When is the final?
Monday, 16 December, 9-11am.
- Can I submit late homework?
No. Instead we drop the 5 lowest grades.
- I missed class. Do you need documentation?

Frequently Asked Questions

From lecture slips & recitation sections.

- When is recitation? It's not on my schedule.
The course is hybrid: 1.5 hours of lecture (T 9:45-11, 118 HN); 0.5 hours quizzes (self-scheduled, 1001E HN); 1.0 hours on-line lab exercises (do anywhere).
- When is the midterm?
There is no midterm. Instead there's required weekly quizzes.
- When is the final?
Monday, 16 December, 9-11am.
- Can I submit late homework?
No. Instead we drop the 5 lowest grades.
- I missed class. Do you need documentation?
*No. Missing lecture & quiz grades are replaced by your final exam score.
If you will miss ≥ 3 weeks ($> 20\%$), see us about taking this in a future term.*

Frequently Asked Questions

From lecture slips & recitation sections.

- When is recitation? It's not on my schedule.
The course is hybrid: 1.5 hours of lecture (T 9:45-11, 118 HN); 0.5 hours quizzes (self-scheduled, 1001E HN); 1.0 hours on-line lab exercises (do anywhere).
- When is the midterm?
There is no midterm. Instead there's required weekly quizzes.
- When is the final?
Monday, 16 December, 9-11am.
- Can I submit late homework?
No. Instead we drop the 5 lowest grades.
- I missed class. Do you need documentation?
No. Missing lecture & quiz grades are replaced by your final exam score. If you will miss ≥ 3 weeks ($> 20\%$), see us about taking this in a future term.
- Can I work ahead?

Frequently Asked Questions

From lecture slips & recitation sections.

- When is recitation? It's not on my schedule.
The course is hybrid: 1.5 hours of lecture (T 9:45-11, 118 HN); 0.5 hours quizzes (self-scheduled, 1001E HN); 1.0 hours on-line lab exercises (do anywhere).
- When is the midterm?
There is no midterm. Instead there's required weekly quizzes.
- When is the final?
Monday, 16 December, 9-11am.
- Can I submit late homework?
No. Instead we drop the 5 lowest grades.
- I missed class. Do you need documentation?
No. Missing lecture & quiz grades are replaced by your final exam score. If you will miss ≥ 3 weeks ($> 20\%$), see us about taking this in a future term.
- Can I work ahead?
Yes! All programs are available, on gradescope, 4 weeks before the deadline.

Frequently Asked Questions

From lecture slips & recitation sections.

- When is recitation? It's not on my schedule.
The course is hybrid: 1.5 hours of lecture (T 9:45-11, 118 HN); 0.5 hours quizzes (self-scheduled, 1001E HN); 1.0 hours on-line lab exercises (do anywhere).
- When is the midterm?
There is no midterm. Instead there's required weekly quizzes.
- When is the final?
Monday, 16 December, 9-11am.
- Can I submit late homework?
No. Instead we drop the 5 lowest grades.
- I missed class. Do you need documentation?
No. Missing lecture & quiz grades are replaced by your final exam score. If you will miss ≥ 3 weeks ($> 20\%$), see us about taking this in a future term.
- Can I work ahead?
Yes! All programs are available, on gradescope, 4 weeks before the deadline.
- Last lecture didn't go into details of programming. Will you in the future?

Frequently Asked Questions

From lecture slips & recitation sections.

- When is recitation? It's not on my schedule.
The course is hybrid: 1.5 hours of lecture (T 9:45-11, 118 HN); 0.5 hours quizzes (self-scheduled, 1001E HN); 1.0 hours on-line lab exercises (do anywhere).
- When is the midterm?
There is no midterm. Instead there's required weekly quizzes.
- When is the final?
Monday, 16 December, 9-11am.
- Can I submit late homework?
No. Instead we drop the 5 lowest grades.
- I missed class. Do you need documentation?
*No. Missing lecture & quiz grades are replaced by your final exam score.
If you will miss ≥ 3 weeks ($> 20\%$), see us about taking this in a future term.*
- Can I work ahead?
Yes! All programs are available, on gradescope, 4 weeks before the deadline.
- Last lecture didn't go into details of programming. Will you in the future?
No worries— today, we'll dive into the details of for, range and string methods.

Frequently Asked Questions

From lecture slips & recitation sections.

- When is recitation? It's not on my schedule.
The course is hybrid: 1.5 hours of lecture (T 9:45-11, 118 HN); 0.5 hours quizzes (self-scheduled, 1001E HN); 1.0 hours on-line lab exercises (do anywhere).
- When is the midterm?
There is no midterm. Instead there's required weekly quizzes.
- When is the final?
Monday, 16 December, 9-11am.
- Can I submit late homework?
No. Instead we drop the 5 lowest grades.
- I missed class. Do you need documentation?
No. Missing lecture & quiz grades are replaced by your final exam score. If you will miss ≥ 3 weeks ($> 20\%$), see us about taking this in a future term.
- Can I work ahead?
Yes! All programs are available, on gradescope, 4 weeks before the deadline.
- Last lecture didn't go into details of programming. Will you in the future?
No worries— today, we'll dive into the details of for, range and string methods.
- You said “when you take second semester...” I just took this class for Pathways...

Frequently Asked Questions

From lecture slips & recitation sections.

- When is recitation? It's not on my schedule.
The course is hybrid: 1.5 hours of lecture (T 9:45-11, 118 HN); 0.5 hours quizzes (self-scheduled, 1001E HN); 1.0 hours on-line lab exercises (do anywhere).
- When is the midterm?
There is no midterm. Instead there's required weekly quizzes.
- When is the final?
Monday, 16 December, 9-11am.
- Can I submit late homework?
No. Instead we drop the 5 lowest grades.
- I missed class. Do you need documentation?
No. Missing lecture & quiz grades are replaced by your final exam score. If you will miss ≥ 3 weeks ($> 20\%$), see us about taking this in a future term.
- Can I work ahead?
Yes! All programs are available, on gradescope, 4 weeks before the deadline.
- Last lecture didn't go into details of programming. Will you in the future?
No worries— today, we'll dive into the details of for, range and string methods.
- You said “when you take second semester...” I just took this class for Pathways...
This is Pathways, but we hope that you will be a CS major/minor.

Frequently Asked Questions

From lecture slips & recitation sections.

- When is recitation? It's not on my schedule.
The course is hybrid: 1.5 hours of lecture (T 9:45-11, 118 HN); 0.5 hours quizzes (self-scheduled, 1001E HN); 1.0 hours on-line lab exercises (do anywhere).
- When is the midterm?
There is no midterm. Instead there's required weekly quizzes.
- When is the final?
Monday, 16 December, 9-11am.
- Can I submit late homework?
No. Instead we drop the 5 lowest grades.
- I missed class. Do you need documentation?
*No. Missing lecture & quiz grades are replaced by your final exam score.
If you will miss ≥ 3 weeks ($> 20\%$), see us about taking this in a future term.*
- Can I work ahead?
Yes! All programs are available, on gradescope, 4 weeks before the deadline.
- Last lecture didn't go into details of programming. Will you in the future?
No worries— today, we'll dive into the details of for, range and string methods.
- You said “when you take second semester...” I just took this class for Pathways...
*This is Pathways, but we hope that you will be a CS major/minor.
We also hope: “Get your education don't forget whence you came...”*

Today's Topics



- Research Survey
- For-loops
- `range()`
- Variables
- Characters
- Strings

Today's Topics



- **Research Survey**
- For-loops
- `range()`
- Variables
- Characters
- Strings

Lecture Slip

Overview

Consent Form

Survey

CSci 127 (Hunter)

Research Study

This study investigates students' emotions, cognitions, motivation, and learning related to computer science.



Part 1: Consists of two brief surveys completed in class.

Prof. John Ranellucci

Educational Psychology

Research Study

This study investigates students' emotions, cognitions, motivation, and learning related to computer science.



Prof. John Ranellucci

Educational Psychology

Part 1: Consists of two brief surveys completed in class.

Part 2: I'm asking you to answer two extra questions at the end of your "lecture slips".

Research Study

This study investigates students' emotions, cognitions, motivation, and learning related to computer science.



Prof. John Ranellucci

Educational Psychology

Part 1: Consists of two brief surveys completed in class.

Part 2: I'm asking you to answer two extra questions at the end of your "lecture slips".

Part 3: Consists of two surveys available online.

Research Study

This study investigates students' emotions, cognitions, motivation, and learning related to computer science.



Prof. John Ranellucci

Educational Psychology

Part 1: Consists of two brief surveys completed in class.

Part 2: I'm asking you to answer two extra questions at the end of your "lecture slips".

Part 3: Consists of two surveys available online.
(Little longer and participants will be compensated with a \$20 Amazon gift certificate for completing both surveys.)

Research Study

This study investigates students' emotions, cognitions, motivation, and learning related to computer science.



Prof. John Ranellucci
Educational Psychology

Part 1: Consists of two brief surveys completed in class.

Part 2: I'm asking you to answer two extra questions at the end of your "lecture slips".

Part 3: Consists of two surveys available online.
(Little longer and participants will be compensated with a \$20 Amazon gift certificate for completing both surveys.)

This study is not part of the class, and no individual analyses will be shared with your instructor. Survey links for the online survey will be emailed to all of you, other surveys will be distributed in class.

Today's Topics



- Research Survey
- **For-loops**
- `range()`
- Variables
- Characters
- Strings

In Pairs or Triples...

Some review and some novel challenges:

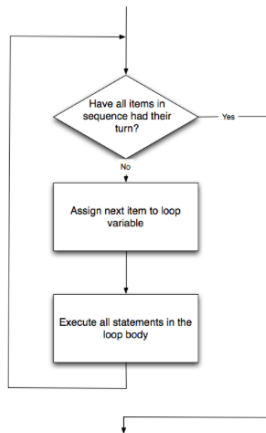
```
1 #Predict what will be printed:
2 for i in range(4):
3     print('The world turned upside down')
4 for j in [0,1,2,3,4,5]:
5     print(j)
6 for count in range(6):
7     print(count)
8 for color in ['red', 'green', 'blue']:
9     print(color)
10 for i in range(2):
11     for j in range(2):
12         print('Look around,')
13     print('How lucky we are to be alive!')
```

Python Tutor

```
1 #Predict what will be printed:
2 for i in range(4):
3     print('The world turned upside down')
4 for j in [0,1,2,3,4,5]:
5     print(j)
6 for count in range(6):
7     print(count)
8 for color in ['red', 'green', 'blue']:
9     print(color)
10 for i in range(2):
11     for j in range(2):
12         print('Look around,')
13     print('How lucky we are to be alive!')
```

(Demo with pythonTutor)

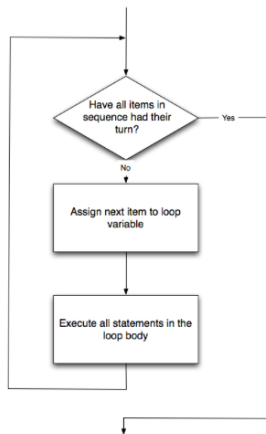
for-loop



```
for i in list:  
    statement1  
    statement2  
    statement3
```

How to Think Like CS, §4.5

for-loop



How to Think Like CS, §4.5

```
for i in list:  
    statement1  
    statement2  
    statement3
```

where `list` is a list of items:

- stated explicitly (e.g. `[1,2,3]`) or
- generated by a function, e.g. `range()`.

In Pairs or Triples...

Some review and some novel challenges:

```
1 #Predict what will be printed:
2
3 for num in [2,4,6,8,10]:
4     print(num)
5
6 sum = 0
7 for x in range(0,12,2):
8     print(x)
9     sum = sum + x
10
11 print(x)
12
13 for c in "ABCD":
14     print(c)
```


Today's Topics



- Research Survey
- For-loops
- **range()**
- Variables
- Characters
- Strings

Python Tutor

```
1 #Predict what will be printed:
2
3 for num in [2,4,6,8,10]:
4     print(num)
5
6 sum = 0
7 for x in range(0,12,2):
8     print(x)
9     sum = sum + x
10
11 print(x)
12
13 for c in "ABCD":
14     print(c)
```

(Demo with pythonTutor)

range()



Simplest version:

- `range(stop)`

range()



Simplest version:

- `range(stop)`
- Produces a list: `[0,1,2,3,...,stop-1]`

range()



Simplest version:

- `range(stop)`
- Produces a list: `[0,1,2,3,...,stop-1]`
- For example, if you want the the list `[0,1,2,3,...,100]`, you would write:

range()



Simplest version:

- `range(stop)`
- Produces a list: `[0,1,2,3,...,stop-1]`
- For example, if you want the the list `[0,1,2,3,...,100]`, you would write:

```
range(101)
```

`range()`

What if you wanted to start somewhere else:



range()

What if you wanted to start somewhere else:

- `range(start, stop)`



range()



What if you wanted to start somewhere else:

- `range(start, stop)`
- Produces a list:
`[start, start+1, ..., stop-1]`

range()



What if you wanted to start somewhere else:

- `range(start, stop)`
- Produces a list:
`[start, start+1, ..., stop-1]`
- For example, if you want the the list
`[10, 11, ..., 20]`
you would write:

range()



What if you wanted to start somewhere else:

- `range(start, stop)`
- Produces a list:
`[start, start+1, ..., stop-1]`
- For example, if you want the the list
`[10, 11, ..., 20]`
you would write:

```
range(10, 21)
```

range()

What if you wanted to count by twos, or some other number:



range()

What if you wanted to count by twos, or some other number:

- `range(start, stop, step)`



range()

What if you wanted to count by twos, or some other number:

- `range(start, stop, step)`
- Produces a list:
`[start, start+step, start+2*step..., last]`
(where last is the largest $start+k*step$ less than stop)



range()

What if you wanted to count by twos, or some other number:

- `range(start, stop, step)`
- Produces a list:
`[start, start+step, start+2*step..., last]`
(where last is the largest $\text{start} + k * \text{step}$ less than stop)
- For example, if you want the the list `[5, 10, ..., 50]` you would write:



range()

What if you wanted to count by twos, or some other number:

- `range(start, stop, step)`
- Produces a list:
`[start, start+step, start+2*step..., last]`
(where last is the largest $\text{start} + k * \text{step}$ less than stop)
- For example, if you want the the list `[5, 10, ..., 50]` you would write:

```
range(5, 51, 5)
```



In summary: `range()`



The three versions:

In summary: `range()`



The three versions:

- `range(stop)`

In summary: `range()`



The three versions:

- `range(stop)`
- `range(start, stop)`

In summary: `range()`



The three versions:

- `range(stop)`
- `range(start, stop)`
- `range(start, stop, step)`

Today's Topics



- Research Survey
- For-loops
- `range()`
- **Variables**
- Characters
- Strings

Variables

- A **variable** is a reserved memory location for storing a value.



Variables



- A **variable** is a reserved memory location for storing a value.
- Different kinds, or **types**, of values need different amounts of space:
 - ▶ **int**: integer or whole numbers

Variables



- A **variable** is a reserved memory location for storing a value.
- Different kinds, or **types**, of values need different amounts of space:
 - ▶ **int**: integer or whole numbers
 - ▶ **float**: floating point or real numbers

Variables



- A **variable** is a reserved memory location for storing a value.
- Different kinds, or **types**, of values need different amounts of space:
 - ▶ **int**: integer or whole numbers
 - ▶ **float**: floating point or real numbers
 - ▶ **string**: sequence of characters

Variables



- A **variable** is a reserved memory location for storing a value.
- Different kinds, or **types**, of values need different amounts of space:
 - ▶ **int**: integer or whole numbers
 - ▶ **float**: floating point or real numbers
 - ▶ **string**: sequence of characters
 - ▶ **list**: a sequence of items

Variables



- A **variable** is a reserved memory location for storing a value.
- Different kinds, or **types**, of values need different amounts of space:
 - ▶ **int**: integer or whole numbers
 - ▶ **float**: floating point or real numbers
 - ▶ **string**: sequence of characters
 - ▶ **list**: a sequence of items
e.g. `[3, 1, 4, 5, 9]` or
`['violet', 'purple', 'indigo']`

Variables



- A **variable** is a reserved memory location for storing a value.
- Different kinds, or **types**, of values need different amounts of space:
 - ▶ **int**: integer or whole numbers
 - ▶ **float**: floating point or real numbers
 - ▶ **string**: sequence of characters
 - ▶ **list**: a sequence of items
e.g. [3, 1, 4, 5, 9] or
['violet', 'purple', 'indigo']
 - ▶ **class variables**: for complex objects, like turtles.

Variable Names

- There's some rules about valid names for variables.



Variable Names



- There's some rules about valid names for variables.
- Can use the underscore ('_'), upper and lower case letters.

Variable Names



- There's some rules about valid names for variables.
- Can use the underscore ('_'), upper and lower case letters.
- Can also use numbers, just can't start a name with a number.

Variable Names



- There's some rules about valid names for variables.
- Can use the underscore ('_'), upper and lower case letters.
- Can also use numbers, just can't start a name with a number.
- Can't use symbols (like '+' or '*') since used for arithmetic.

Variable Names



- There's some rules about valid names for variables.
- Can use the underscore ('_'), upper and lower case letters.
- Can also use numbers, just can't start a name with a number.
- Can't use symbols (like '+' or '*') since used for arithmetic.
- Can't use some words that Python has reserved for itself (e.g. `for`).
(List of reserved words in *Think CS*, §2.5.)

Today's Topics



- Research Survey
- For-loops
- `range()`
- Variables
- **Characters**
- Strings

Standardized Code for Characters

American Standard Code for Information Interchange (ASCII), 1960.

Standardized Code for Characters

American Standard Code for Information Interchange (ASCII), 1960.
(New version called: Unicode).

Standardized Code for Characters

American Standard Code for Information Interchange (ASCII), 1960.

(New version called: Unicode).

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	\$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

(wiki)

Converting from Character to Code:

(There is an ASCII table on the back of today's lecture slip.)

ASCII TABLE

Decimal	Hex Char	Decimal	Hex Char	Decimal	Hex Char	Decimal	Hex Char
0		16	P	32	@	48	0
1	!	17	Q	33	A	49	1
2	"	18	R	34	B	50	2
3	#	19	S	35	C	51	3
4	\$	20	T	36	D	52	4
5	%	21	U	37	E	53	5
6	&	22	V	38	F	54	6
7	'	23	W	39	G	55	7
8	(24	X	40	H	56	8
9)	25	Y	41	I	57	9
10	*	26	Z	42	J	58	.
11	+	27	[43	K	59	,
12	,	28	\	44	L	60	:
13	-	29]	45	M	61	;
14	.	30	^	46	N	62	'
15	/	31	_	47	O	63	~
16				48	P		
17				49	Q		
18				50	R		
19				51	S		
20				52	T		
21				53	U		
22				54	V		
23				55	W		
24				56	X		
25				57	Y		
26				58	Z		
27				59	[
28				60	\		
29				61]		
30				62	^		
31				63	_		
32	@	64	0	96	0	128	
33	A	65	1	97	a	129	
34	B	66	2	98	b	130	
35	C	67	3	99	c	131	
36	D	68	4	100	d	132	
37	E	69	5	101	e	133	
38	F	70	6	102	f	134	
39	G	71	7	103	g	135	
40	H	72	8	104	h	136	
41	I	73	9	105	i	137	
42	J	74	A	106	j	138	
43	K	75	B	107	k	139	
44	L	76	C	108	l	140	
45	M	77	D	109	m	141	
46	N	78	E	110	n	142	
47	O	79	F	111	o	143	
48	0	80	G	112	p	144	
49	1	81	H	113	q	145	
50	2	82	I	114	r	146	
51	3	83	J	115	s	147	
52	4	84	K	116	t	148	
53	5	85	L	117	u	149	
54	6	86	M	118	v	150	
55	7	87	N	119	w	151	
56	8	88	O	120	x	152	
57	9	89	P	121	y	153	
58	.	90	Q	122	z	154	
59	,	91	R	123	[155	
60	:	92	S	124	\	156	
61	;	93	T	125]	157	
62	'	94	U	126	^	158	
63	~	95	V	127	_	159	
64		96		128		160	
65		97		129		161	
66		98		130		162	
67		99		131		163	
68		100		132		164	
69		101		133		165	
70		102		134		166	
71		103		135		167	
72		104		136		168	
73		105		137		169	
74		106		138		170	
75		107		139		171	
76		108		140		172	
77		109		141		173	
78		110		142		174	
79		111		143		175	
80		112		144		176	
81		113		145		177	
82		114		146		178	
83		115		147		179	
84		116		148		180	
85		117		149		181	
86		118		150		182	
87		119		151		183	
88		120		152		184	
89		121		153		185	
90		122		154		186	
91		123		155		187	
92		124		156		188	
93		125		157		189	
94		126		158		190	
95		127		159		191	
96		128		160		192	
97		129		161		193	
98		130		162		194	
99		131		163		195	
100		132		164		196	
101		133		165		197	
102		134		166		198	
103		135		167		199	
104		136		168		200	
105		137		169		201	
106		138		170		202	
107		139		171		203	
108		140		172		204	
109		141		173		205	
110		142		174		206	
111		143		175		207	
112		144		176		208	
113		145		177		209	
114		146		178		210	
115		147		179		211	
116		148		180		212	
117		149		181		213	
118		150		182		214	
119		151		183		215	
120		152		184		216	
121		153		185		217	
122		154		186		218	
123		155		187		219	
124		156		188		220	
125		157		189		221	
126		158		190		222	
127		159		191		223	
128		160		192		224	
129		161		193		225	
130		162		194		226	
131		163		195		227	
132		164		196		228	
133		165		197		229	
134		166		198		230	
135		167		199		231	
136		168		200		232	
137		169		201		233	
138		170		202		234	
139		171		203		235	
140		172		204		236	
141		173		205		237	
142		174		206		238	
143		175		207		239	
144		176		208		240	
145		177		209		241	
146		178		210		242	
147		179		211		243	
148		180		212		244	
149		181		213		245	
150		182		214		246	
151		183		215		247	
152		184		216		248	
153		185		217		249	
154		186		218		250	
155		187		219		251	
156		188		220		252	
157		189		221		253	
158		190		222		254	
159		191		223		255	
160		192		224			
161		193		225			
162		194		226			
163		195		227			
164		196		228			
165		197		229			
166		198		230			
167		199		231			
168		200		232			
169		201		233			
170		202		234			
171		203		235			
172		204		236			
173		205		237			
174		206		238			
175		207		239			
176		208		240			
177		209		241			
178		210		242			
179		211		243			
180		212		244			
181		213		245			
182		214		246			
183		215		247			
184		216		248			
185		217		249			
186		218		250			
187		219		251			
188		220		252			
189		221		253			
190		222		254			
191		223		255			
192		224					
193		225					
194		226					
195		227					
196		228					
197		229					
198		230					
199		231					
200		232					
201		233					
202		234					
203		235					
204		236					
205		237					
206		238					
207		239					
208		240					
209		241					
210		242					
211		243					
212		244					
213		245					
214		246					
215		247					
216		248					
217		249					
218		250					
219		251					
220		252					
221		253					
222		254					
223		255					
224							
225							
226							
227							
228							
229							
230							
231							
232							
233							
234							
235							
236							
237							
238							
239							
240							
241							
242							
243							
244							
245							
246							
247							
248							
249							

Converting from Character to Code:

(There is an ASCII table on the back of today's lecture slip.)

- `ord(c)`: returns Unicode (ASCII) of the character.

ASCII TABLE

Decimal	Hex Char	Decimal	Hex Char	Decimal	Hex Char	Decimal	Hex Char
0		16	0	32	@	48	0
1		17	1	33	!	49	1
2		18	2	34	"	50	2
3		19	3	35	#	51	3
4		20	4	36	\$	52	4
5		21	5	37	%	53	5
6		22	6	38	&	54	6
7		23	7	39	'	55	7
8		24	8	40	(56	8
9		25	9	41)	57	9
10		26	A	42	*	58	A
11		27	B	43	+	59	B
12		28	C	44	,	60	C
13		29	D	45	-	61	D
14		30	E	46	.	62	E
15		31	F	47	/	63	F
16	0	32	@	64	0	80	0
17	1	33	!	65	A	81	1
18	2	34	"	66	B	82	2
19	3	35	#	67	C	83	3
20	4	36	\$	68	D	84	4
21	5	37	%	69	E	85	5
22	6	38	&	70	F	86	6
23	7	39	'	71		87	7
24	8	40	(72	a	88	8
25	9	41)	73	b	89	9
26	A	42	*	74	c	90	A
27	B	43	+	75	d	91	B
28	C	44	,	76	e	92	C
29	D	45	-	77	f	93	D
30	E	46	.	78	g	94	E
31	F	47	/	79	h	95	F
32		48	0	80	i	96	
33	!	49	1	81	j	97	a
34	"	50	2	82	k	98	b
35	#	51	3	83	l	99	c
36	\$	52	4	84	m	100	d
37	%	53	5	85	n	101	e
38	&	54	6	86	o	102	f
39	'	55	7	87	p	103	g
40	(56	8	88	q	104	h
41)	57	9	89	r	105	i
42	*	58	A	90	s	106	j
43	+	59	B	91	t	107	k
44	,	60	C	92	u	108	l
45	-	61	D	93	v	109	m
46	.	62	E	94	w	110	n
47	/	63	F	95	x	111	o
48	0	64		96	y	112	p
49	1	65	A	97	z	113	q
50	2	66	B	98		114	r
51	3	67	C	99		115	s
52	4	68	D	100		116	t
53	5	69	E	101		117	u
54	6	70	F	102		118	v
55	7	71		103		119	w
56	8	72	a	104		120	x
57	9	73	b	105		121	y
58	A	74	c	106		122	z
59	B	75	d	107		123	
60	C	76	e	108		124	
61	D	77	f	109		125	
62	E	78	g	110		126	
63	F	79	h	111		127	
64		80	i	112		128	
65	A	81	j	113		129	
66	B	82	k	114		130	
67	C	83	l	115		131	
68	D	84	m	116		132	
69	E	85	n	117		133	
70	F	86	o	118		134	
71		87	p	119		135	
72	a	88	q	120		136	
73	b	89	r	121		137	
74	c	90	s	122		138	
75	d	91	t	123		139	
76	e	92	u	124		140	
77	f	93	v	125		141	
78	g	94	w	126		142	
79	h	95	x	127		143	
80	i	96	y	128		144	
81	j	97	z	129		145	
82		98		130		146	
83		99		131		147	
84		100		132		148	
85		101		133		149	
86		102		134		150	
87		103		135		151	
88		104		136		152	
89		105		137		153	
90		106		138		154	
91		107		139		155	
92		108		140		156	
93		109		141		157	
94		110		142		158	
95		111		143		159	
96		112		144		160	
97		113		145		161	
98		114		146		162	
99		115		147		163	
100		116		148		164	
101		117		149		165	
102		118		150		166	
103		119		151		167	
104		120		152		168	
105		121		153		169	
106		122		154		170	
107		123		155		171	
108		124		156		172	
109		125		157		173	
110		126		158		174	
111		127		159		175	
112		128		160		176	
113		129		161		177	
114		130		162		178	
115		131		163		179	
116		132		164		180	
117		133		165		181	
118		134		166		182	
119		135		167		183	
120		136		168		184	
121		137		169		185	
122		138		170		186	
123		139		171		187	
124		140		172		188	
125		141		173		189	
126		142		174		190	
127		143		175		191	
128		144		176		192	
129		145		177		193	
130		146		178		194	
131		147		179		195	
132		148		180		196	
133		149		181		197	
134		150		182		198	
135		151		183		199	
136		152		184		200	
137		153		185		201	
138		154		186		202	
139		155		187		203	
140		156		188		204	
141		157		189		205	
142		158		190		206	
143		159		191		207	
144		160		192		208	
145		161		193		209	
146		162		194		210	
147		163		195		211	
148		164		196		212	
149		165		197		213	
150		166		198		214	
151		167		199		215	
152		168		200		216	
153		169		201		217	
154		170		202		218	
155		171		203		219	
156		172		204		220	
157		173		205		221	
158		174		206		222	
159		175		207		223	
160		176		208		224	
161		177		209		225	
162		178		210		226	
163		179		211		227	
164		180		212		228	
165		181		213		229	
166		182		214		230	
167		183		215		231	
168		184		216		232	
169		185		217		233	
170		186		218		234	
171		187		219		235	
172		188		220		236	
173		189		221		237	
174		190		222		238	
175		191		223		239	
176		192		224		240	
177		193		225		241	
178		194		226		242	
179		195		227		243	
180		196		228		244	
181		197		229		245	
182		198		230		246	
183		199		231		247	
184		200		232		248	
185		201		233		249	
186		202		234		250	
187		203		235		251	
188		204		236		252	
189		205		237		253	
190		206		238		254	
191		207		239		255	
192		208		240		256	
193		209		241		257	
194		210		242		258	
195		211		243		259	
196		212		244		260	
197		213		245		261	
198		214		246		262	
199		215		247		263	
200		216		248		264	
201		217		249		265	
202		218		250		266	
203		219		251		267	
204		220		252		268	
205		221		253		269	
206		222		254		270	
207		223		255		271	
208		224		256		272	
209		225		257		273	
210		226		258		274	
211		227		259		275	
212		228		260		276	
213		229		261		277	
214		230		262		278	
215		231		263		279	
216		232		264		280	
217		233		265		281	
218		234		266		282	
219		235		267		283	
220		236		268		284	
221		237		269		285	
222		238		270		286	
223		239		271		287	
224		240		272		288	
225		241		273		289	
226		242		274		290	
227		243		275		291	
228		244		276		292	
229		245		277		293	
230		246		278		294	
231		247		279		295	
23							

Converting from Character to Code:

(There is an ASCII table on the back of today's lecture slip.)

ASCII TABLE

Decimal	Hex Char	Decimal	Hex Char	Decimal	Hex Char	Decimal	Hex Char
0		16		32		48	
1		17		33	!	49	1
2		18		34	"	50	2
3		19		35	#	51	3
4		20		36	\$	52	4
5		21		37	%	53	5
6		22		38	&	54	6
7		23		39	'	55	7
8		24		40	(56	8
9		25		41)	57	9
10		26		42	*	58	:
11		27		43	+	59	;
12		28		44	,	60	<
13		29		45	-	61	=
14		30		46	.	62	>
15		31		47	/	63	?
16		32	!	64	@	80	P
17		33	"	65	A	81	Q
18		34	"	66	B	82	R
19		35	#	67	C	83	S
20		36	\$	68	D	84	T
21		37	%	69	E	85	U
22		38	&	70	F	86	V
23		39	'	71	G	87	W
24		40	(72	H	88	X
25		41)	73	I	89	Y
26		42	*	74	J	90	Z
27		43	+	75	K	91	[
28		44	,	76	L	92	\
29		45	-	77	M	93]
30		46	.	78	N	94	^
31		47	/	79	O	95	_
32	!	48	0	80	P	96	`
33	"	49	1	81	Q	97	a
34	"	50	2	82	R	98	b
35	#	51	3	83	S	99	c
36	\$	52	4	84	T	100	d
37	%	53	5	85	U	101	e
38	&	54	6	86	V	102	f
39	'	55	7	87	W	103	g
40	(56	8	88	X	104	h
41)	57	9	89	Y	105	i
42	*	58	:	90	Z	106	j
43	+	59	;	91	[107	k
44	,	60	<	92	\	108	l
45	-	61	=	93]	109	m
46	.	62	>	94	^	110	n
47	/	63	?	95	_	111	o
48	0	64	@	96	`	112	p
49	1	65	A	97	a	113	q
50	2	66	B	98	b	114	r
51	3	67	C	99	c	115	s
52	4	70	F	100	d	116	t
53	5	71	G	101	e	117	u
54	6	72	H	102	f	118	v
55	7	73	I	103	g	119	w
56	8	74	J	104	h	120	x
57	9	75	K	105	i	121	y
58	:	76	L	106	j	122	z
59	;	77	M	107	k	123	{
60	<	78	N	108	l	124	
61	=	79	O	109	m	125	}
62	>	80	P	110	n	126	~
63	?	81	Q	111	o	127	
64	@	82	R	112	p		
65	A	83	S	113	q		
66	B	84	T	114	r		
67	C	85	U	115	s		
68	D	86	V	116	t		
69	E	87	W	117	u		
70	F	88	X	118	v		
71	G	89	Y	119	w		
72	H	90	Z	120	x		
73	I	91	[121	y		
74	J	92	\	122	z		
75	K	93]				
76	L	94	^				
77	M	95	_				
78	N	96	`				
79	O	97	a				
80	P	98	b				
81	Q	99	c				
82	R	100	d				
83	S	101	e				
84	T	102	f				
85	U	103	g				
86	V	104	h				
87	W	105	i				
88	X	106	j				
89	Y	107	k				
90	Z	108	l				
91	[109	m				
92	\	110	n				
93]	111	o				
94	^	112	p				
95	_	113	q				
96	`	114	r				
97	a	115	s				
98	b	116	t				
99	c	117	u				
100	d	118	v				
101	e	119	w				
102	f	120	x				
103	g	121	y				
104	h	122	z				
105	i						
106	j						
107	k						
108	l						
109	m						
110	n						
111	o						
112	p						
113	q						
114	r						
115	s						
116	t						
117	u						
118	v						
119	w						
120	x						
121	y						
122	z						
123	{						
124							
125	}						
126	~						
127							

- `ord(c)`: returns Unicode (ASCII) of the character.
- Example: `ord('a')` returns 97.

Converting from Character to Code:

(There is an ASCII table on the back of today's lecture slip.)

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	00		16	10	P	32	20	[48	30	0
1	01	SOH	17	11	Q	33	21	\	49	31	1
2	02	STX	18	12	R	34	22]	50	32	2
3	03	ETX	19	13	S	35	23	^	51	33	3
4	04	END	20	14	T	36	24	_	52	34	4
5	05	SO	21	15	U	37	25	`	53	35	5
6	06	SI	22	16	V	38	26	{	54	36	6
7	07	DEL	23	17	W	39	27		55	37	7
8	08		24	18	X	40	28	~	56	38	8
9	09	LF	25	19	Y	41	29		57	39	9
10	0A	LF	26	1A	Z	42	2A	DEL	58	3A	:
11	0B	VT	27	1B	[43	2B		59	3B	;
12	0C	FF	28	1C	\	44	2C		60	3C	<
13	0D	CR	29	1D]	45	2D		61	3D	=
14	0E		30	1E	^	46	2E		62	3E	>
15	0F		31	1F	_	47	2F		63	3F	?
16	10	@	32	20	[48	30	0	64	40	@
17	11	A	33	21	\	49	31	1	65	41	A
18	12	B	34	22]	50	32	2	66	42	B
19	13	C	35	23	^	51	33	3	67	43	C
20	14	D	36	24	_	52	34	4	68	44	D
21	15	E	37	25	`	53	35	5	69	45	E
22	16	F	38	26	{	54	36	6	70	46	F
23	17	G	39	27		55	37	7	71	47	G
24	18	H	40	28	~	56	38	8	72	48	H
25	19	I	41	29		57	39	9	73	49	I
26	1A	J	42	2A	DEL	58	3A	:	74	4A	J
27	1B	K	43	2B		59	3B	;	75	4B	K
28	1C	L	44	2C		60	3C	<	76	4C	L
29	1D	M	45	2D		61	3D	=	77	4D	M
30	1E	N	46	2E		62	3E	>	78	4E	N
31	1F	O	47	2F		63	3F	?	79	4F	O
32	20	SP	48	30	0	64	40	@	80	50	P
33	21	P	49	31	1	65	41	A	81	51	Q
34	22	Q	50	32	2	66	42	B	82	52	R
35	23	R	51	33	3	67	43	C	83	53	S
36	24	S	52	34	4	68	44	D	84	54	T
37	25	T	53	35	5	69	45	E	85	55	U
38	26	U	54	36	6	70	46	F	86	56	V
39	27	V	55	37	7	71	47	G	87	57	W
40	28	W	56	38	8	72	48	H	88	58	X
41	29	X	57	39	9	73	49	I	89	59	Y
42	2A	Y	58	3A	:	74	4A	J	90	5A	Z
43	2B	Z	59	3B	;	75	4B	K			
44	2C	[60	3C	<	76	4C	L			
45	2D	\	61	3D	=	77	4D	M			
46	2E]	62	3E	>	78	4E	N			
47	2F	^	63	3F	?	79	4F	O			
48	30	_	64	40	@	80	50	P			
49	31	`	65	41	A	81	51	Q			
50	32	{	66	42	B	82	52	R			
51	33		67	43	C	83	53	S			
52	34	~	68	44	D	84	54	T			
53	35		69	45	E	85	55	U			
54	36		70	46	F	86	56	V			
55	37		71	47	G	87	57	W			
56	38		72	48	H	88	58	X			
57	39		73	49	I	89	59	Y			
58	3A		74	4A	J	90	5A	Z			
59	3B		75	4B	K						
60	3C		76	4C	L						
61	3D		77	4D	M						
62	3E		78	4E	N						
63	3F		79	4F	O						
64	40	@	80	50	P						
65	41	A	81	51	Q						
66	42	B	82	52	R						
67	43	C	83	53	S						
68	44	D	84	54	T						
69	45	E	85	55	U						
70	46	F	86	56	V						
71	47	G	87	57	W						
72	48	H	88	58	X						
73	49	I	89	59	Y						
74	4A	J	90	5A	Z						
75	4B	K									
76	4C	L									
77	4D	M									
78	4E	N									
79	4F	O									
80	50	P									
81	51	Q									
82	52	R									
83	53	S									
84	54	T									
85	55	U									
86	56	V									
87	57	W									
88	58	X									
89	59	Y									
90	5A	Z									
91	5B	[
92	5C	\									
93	5D]									
94	5E	^									
95	5F	_									
96	60	`									
97	61	{									
98	62										
99	63	~									
100	64										

- `ord(c)`: returns Unicode (ASCII) of the character.
- Example: `ord('a')` returns 97.
- `chr(x)`: returns the character whose Unicode is x.

Converting from Character to Code:

(There is an ASCII table on the back of today's lecture slip.)

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	00		16	10	P	32	20	[48	30	0
1	01		17	11	Q	33	21	\	49	31	1
2	02		18	12	R	34	22]	50	32	2
3	03		19	13	S	35	23	^	51	33	3
4	04		20	14	T	36	24	_	52	34	4
5	05		21	15	U	37	25	`	53	35	5
6	06		22	16	V	38	26	{	54	36	6
7	07		23	17	W	39	27		55	37	7
8	08		24	18	X	40	28	~	56	38	8
9	09		25	19	Y	41	29		57	39	9
10	0A		26	1A	Z	42	2A		58	3A	.
11	0B		27	1B	[43	2B		59	3B	,
12	0C		28	1C	\	44	2C		60	3C	;
13	0D		29	1D]	45	2D		61	3D	'
14	0E		30	1E	^	46	2E		62	3E	"
15	0F		31	1F	_	47	2F		63	3F	!
16	10	@	32	20	[48	30	0	64	40	+
17	11	A	33	21	\	49	31	1	65	41	*
18	12	B	34	22]	50	32	2	66	42	-
19	13	C	35	23	^	51	33	3	67	43	=
20	14	D	36	24	_	52	34	4	68	44	&
21	15	E	37	25	`	53	35	5	69	45	%
22	16	F	38	26	{	54	36	6	70	46	@
23	17		39	27		55	37	7	71	47	#
24	18		40	28	~	56	38	8	72	48	\$
25	19		41	29		57	39	9	73	49	%
26	1A		42	2A		58	3A	.	74	4A	&
27	1B		43	2B		59	3B	,	75	4B	'
28	1C		44	2C		60	3C	;	76	4C	"
29	1D		45	2D		61	3D	'	77	4D	!
30	1E		46	2E		62	3E	"	78	4E	+
31	1F		47	2F		63	3F	!	79	4F	*
32	20		48	30	0	64	40	+	80	50	^
33	21	!	49	31	1	65	41	*	81	51	_
34	22	"	50	32	2	66	42	-	82	52	~
35	23	"	51	33	3	67	43	=	83	53	~
36	24	\$	52	34	4	68	44	&	84	54	~
37	25	%	53	35	5	69	45	%	85	55	~
38	26	&	54	36	6	70	46	@	86	56	~
39	27	'	55	37	7	71	47	#	87	57	~
40	28	(56	38	8	72	48	\$	88	58	~
41	29)	57	39	9	73	49	%	89	59	~
42	2A	*	58	3A	.	74	4A	&	90	5A	~
43	2B	+	59	3B	,	75	4B	'	91	5B	~
44	2C	,	60	3C	;	76	4C	"	92	5C	~
45	2D	-	61	3D	'	77	4D	!	93	5D	~
46	2E	.	62	3E	"	78	4E	+	94	5E	~
47	2F	/	63	3F	!	79	4F	*	95	5F	~
48	30	0	64	40	+	80	50	^	96	60	~
49	31	1	65	41	*	81	51	_	97	61	~
50	32	2	66	42	-	82	52	~	98	62	~
51	33	3	67	43	=	83	53	~	99	63	~
52	34	4	68	44	&	84	54	~	100	64	~
53	35	5	69	45	%	85	55	~			
54	36	6	70	46	@	86	56	~			
55	37	7	71	47	#	87	57	~			
56	38	8	72	48	\$	88	58	~			
57	39	9	73	49	%	89	59	~			
58	3A	.	74	4A	&	90	5A	~			
59	3B	,	75	4B	'	91	5B	~			
60	3C	;	76	4C	"	92	5C	~			
61	3D	'	77	4D	!	93	5D	~			
62	3E	"	78	4E	+	94	5E	~			
63	3F	!	79	4F	*	95	5F	~			
64	40	+	80	50	^	96	60	~			
65	41	*	81	51	_	97	61	~			
66	42	-	82	52	~	98	62	~			
67	43	=	83	53	~	99	63	~			
68	44	&	84	54	~						
69	45	%	85	55	~						
70	46	@	86	56	~						
71	47	#	87	57	~						
72	48	\$	88	58	~						
73	49	%	89	59	~						
74	4A	&	90	5A	~						
75	4B	'	91	5B	~						
76	4C	"	92	5C	~						
77	4D	!	93	5D	~						
78	4E	+	94	5E	~						
79	4F	*	95	5F	~						
80	50	^	96	60	~						
81	51	_	97	61	~						
82	52	~	98	62	~						
83	53	~	99	63	~						
84	54	~									
85	55	~									
86	56	~									
87	57	~									
88	58	~									
89	59	~									
90	5A	~									
91	5B	~									
92	5C	~									
93	5D	~									
94	5E	~									
95	5F	~									
96	60	~									
97	61	~									
98	62	~									
99	63	~									
100	64	~									

- `ord(c)`: returns Unicode (ASCII) of the character.
- Example: `ord('a')` returns 97.
- `chr(x)`: returns the character whose Unicode is x.
- Example: `chr(97)` returns 'a'.

Converting from Character to Code:

(There is an ASCII table on the back of today's lecture slip.)

ASCII TABLE

Decimal	Hex Char	Decimal	Hex Char	Decimal	Hex Char	Decimal	Hex Char
0		16	P	32	@	48	0
1		17	Q	33	A	49	1
2		18	R	34	B	50	2
3		19	S	35	C	51	3
4		20	T	36	D	52	4
5		21	U	37	E	53	5
6		22	V	38	F	54	6
7		23	W	39	G	55	7
8		24	X	40	H	56	8
9		25	Y	41	I	57	9
10		26	Z	42	J	58	.
11		27	[43	K	59	,
12		28	\	44	L	60	:
13		29]	45	M	61	;
14		30	^	46	N	62	'
15		31	_	47	O	63	"
16	P	32	@	48	0	64	~
17	Q	33	A	49	1	65	~
18	R	34	B	50	2	66	~
19	S	35	C	51	3	67	~
20	T	36	D	52	4	68	~
21	U	37	E	53	5	69	~
22	V	38	F	54	6	70	~
23	W	39	G	55	7	71	~
24	X	40	H	56	8	72	~
25	Y	41	I	57	9	73	~
26	Z	42	J	58	.	74	~
27	[43	K	59	,	75	~
28	\	44	L	60	:	76	~
29]	45	M	61	;	77	~
30	^	46	N	62	'	78	~
31	_	47	O	63	"	79	~
32	@	48	0	64	~	80	~
33	A	49	1	65	~	81	~
34	B	50	2	66	~	82	~
35	C	51	3	67	~	83	~
36	D	52	4	68	~	84	~
37	E	53	5	69	~	85	~
38	F	54	6	70	~	86	~
39	G	55	7	71	~	87	~
40	H	56	8	72	~	88	~
41	I	57	9	73	~	89	~
42	J	58	.	74	~	90	~
43	K	59	,	75	~	91	~
44	L	60	:	76	~	92	~
45	M	61	;	77	~	93	~
46	N	62	'	78	~	94	~
47	O	63	"	79	~	95	~
48	0	64	~	80	~	96	~
49	1	65	~	81	~	97	~
50	2	66	~	82	~	98	~
51	3	67	~	83	~	99	~
52	4	68	~	84	~	100	~
53	5	69	~	85	~	101	~
54	6	70	~	86	~	102	~
55	7	71	~	87	~	103	~
56	8	72	~	88	~	104	~
57	9	73	~	89	~	105	~
58	.	74	~	90	~	106	~
59	,	75	~	91	~	107	~
60	:	76	~	92	~	108	~
61	;	77	~	93	~	109	~
62	'	78	~	94	~	110	~
63	"	79	~	95	~	111	~
64	~	80	~	96	~	112	~
65	~	81	~	97	~	113	~
66	~	82	~	98	~	114	~
67	~	83	~	99	~	115	~
68	~	84	~	100	~	116	~
69	~	85	~	101	~	117	~
70	~	86	~	102	~	118	~
71	~	87	~	103	~	119	~
72	~	88	~	104	~	120	~
73	~	89	~	105	~	121	~
74	~	90	~	106	~	122	~
75	~	91	~	107	~	123	~
76	~	92	~	108	~	124	~
77	~	93	~	109	~	125	~
78	~	94	~	110	~	126	~
79	~	95	~	111	~	127	DEL

- `ord(c)`: returns Unicode (ASCII) of the character.
- Example: `ord('a')` returns 97.
- `chr(x)`: returns the character whose Unicode is x.
- Example: `chr(97)` returns 'a'.
- What is `chr(33)`?

In Pairs or Triples...

Some review and some novel challenges:

```
1 #Predict what will be printed:
2
3 for c in range(65,90):
4     print(chr(c))
5
6 message = "I love Python"
7 newMessage = ""
8 for c in message:
9     print(ord(c))    #Print the Unicode of each number
10    print(chr(ord(c)+1))    #Print the next character
11    newMessage = newMessage + chr(ord(c)+1) #add to the new message
12 print("The coded message is", newMessage)
13
14 word = "zebra"
15 codedWord = ""
16 for ch in word:
17     offset = ord(ch) - ord('a') + 1 #how many letters past 'a'
18     wrap = offset % 26 #if larger than 26, wrap back to 0
19     newChar = chr(ord('a') + wrap) #compute the new letter
20     print(wrap, chr(ord('a') + wrap))    #print the wrap & new lett
21     codedWord = codedWord + newChar #add the newChar to the coded w
22
23 print("The coded word (with wrap) is", codedWord)
```

Python Tutor

```
1 #Predict what will be printed:
2
3 for c in range(65,90):
4     print(chr(c))
5
6 message = "I love Python"
7 newMessage = ""
8 for c in message:
9     print(ord(c))    #Print the Unicode of each number
10    print(chr(ord(c)+1))    #Print the next character
11    newMessage = newMessage + chr(ord(c)+1) #Add to the new message
12 print("The coded message is", newMessage)
13
14 word = "zebra"
15 codedWord = ""
16 for ch in word:
17     offset = ord(ch) - ord('a') + 1 #how many letters past 'a'
18     wrap = offset % 26 #if larger than 26, wrap back to 0
19     newChar = chr(ord('a') + wrap) #compute the new letter
20     print(wrap, chr(ord('a') + wrap))    #Print the wrap & new lett
21     codedWord = codedWord + newChar #add the newChar to the coded w
22
23 print("The coded word (with wrap) is", codedWord)
```

(Demo with pythonTutor)

User Input

Covered in detail in Lab 2:

```
➔ 1 mess = input('Please enter a message: ')\n  2 print("You entered", mess)
```

(Demo with pythonTutor)

Side Note: '+' for numbers and strings



- `x = 3 + 5` stores the number 8 in memory location `x`.

Side Note: '+' for numbers and strings



- $x = 3 + 5$ stores the number 8 in memory location x .
- $x = x + 1$ increases x by 1.

Side Note: '+' for numbers and strings



- `x = 3 + 5` stores the number 8 in memory location `x`.
- `x = x + 1` increases `x` by 1.
- `s = "hi" + "Mom"` stores "hiMom" in memory locations `s`.

Side Note: '+' for numbers and strings



- `x = 3 + 5` stores the number 8 in memory location `x`.
- `x = x + 1` increases `x` by 1.
- `s = "hi" + "Mom"` stores "hiMom" in memory locations `s`.
- `s = s + "A"` adds the letter "A" to the end of the strings `s`.

Today's Topics



- Research Survey
- For-loops
- `range()`
- Variables
- Characters
- **Strings**

More on Strings...

From Final Exam, Fall 2017, Version 1, #1:

Name:

EmpID:

CSci 127 Final, V1, F17

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[:-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:

More on Strings...

Name:

EmpID:

CSci 127 Final, V1, F17

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[:-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:



- Some we have seen before, some we haven't.

More on Strings...

Name:

EmpID:

CSci 127 Final, V1, F17

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[:-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:



- Some we have seen before, some we haven't.
- Don't leave it blank— write what you know & puzzle out as much as possible.

More on Strings...

Name:

EmpID:

CSci 127 Final, V1, F17

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[:-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:



- Some we have seen before, some we haven't.
- Don't leave it blank— write what you know & puzzle out as much as possible.
- First, go through and write down what we know:

More on Strings...

Name:

EmpID:

CSci 127 Final, V1, F17

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[:-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:



- Some we have seen before, some we haven't.
- Don't leave it blank— write what you know & puzzle out as much as possible.
- First, go through and write down what we know:
 - ▶ There are 3 print().

More on Strings...

Name:

EmpID:

CSci 127 Final, V1, F17

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[:-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:



- Some we have seen before, some we haven't.
- Don't leave it blank— write what you know & puzzle out as much as possible.
- First, go through and write down what we know:
 - ▶ There are 3 `print()`.
 - ▶ Output will have at least:

More on Strings...

Name:

EmpID:

CSci 127 Final, V1, F17

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[:-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:

- Some we have seen before, some we haven't.
- Don't leave it blank— write what you know & puzzle out as much as possible.
- First, go through and write down what we know:
 - There are 3 `print()`.
 - Output will have at least:
There are ??? fun days in a week

More on Strings...

Name:

EmpID:

CSci 127 Final, V1, F17

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[:-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:



- Some we have seen before, some we haven't.
- Don't leave it blank— write what you know & puzzle out as much as possible.
- First, go through and write down what we know:
 - ▶ There are 3 `print()`.
 - ▶ Output will have at least:
There are ??? fun days in a week
Two of them are ???

More on Strings...

Name:

EmpID:

CSci 127 Final, V1, F17

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[:-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:



- Some we have seen before, some we haven't.
- Don't leave it blank— write what you know & puzzle out as much as possible.
- First, go through and write down what we know:
 - ▶ There are 3 print().
 - ▶ Output will have at least:
There are ??? fun days in a week
Two of them are ???
My favorite ??? is Saturday.

More on Strings...

Name:

EmpID:

CSci 127 Final, V1, F17

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[:-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:

- Some we have seen before, some we haven't.
- Don't leave it blank— write what you know & puzzle out as much as possible.
- First, go through and write down what we know:
 - There are 3 print().
 - Output will have at least:
There are ??? fun days in a week
Two of them are ???
My favorite ??? is Saturday.
- Will get 1/3 to 1/2 points for writing down the basic structure.

More on Strings: String Methods

```
s = "FridaysSaturdaysSundays"  
num = s.count("s")
```

- The first line creates a variable, called `s`, that stores the string: `"FridaysSaturdaysSundays"`

More on Strings: String Methods

```
s = "FridaysSaturdaysSundays"  
num = s.count("s")
```

- The first line creates a variable, called `s`, that stores the string: `"FridaysSaturdaysSundays"`
- There are many useful functions for strings (more in Lab 2).

More on Strings: String Methods

```
s = "FridaysSaturdaysSundays"  
num = s.count("s")
```

- The first line creates a variable, called `s`, that stores the string: `"FridaysSaturdaysSundays"`
- There are many useful functions for strings (more in Lab 2).
- `s.count(x)` will count the number of times the pattern, `x`, appears in `s`.

More on Strings: String Methods

```
s = "FridaysSaturdaysSundays"  
num = s.count("s")
```

- The first line creates a variable, called `s`, that stores the string: `"FridaysSaturdaysSundays"`
- There are many useful functions for strings (more in Lab 2).
- `s.count(x)` will count the number of times the pattern, `x`, appears in `s`.
 - ▶ `s.count("s")` counts the number of lower case `s` that occurs.

More on Strings: String Methods

```
s = "FridaysSaturdaysSundays"  
num = s.count("s")
```

- The first line creates a variable, called `s`, that stores the string: `"FridaysSaturdaysSundays"`
- There are many useful functions for strings (more in Lab 2).
- `s.count(x)` will count the number of times the pattern, `x`, appears in `s`.
 - ▶ `s.count("s")` counts the number of lower case `s` that occurs.
 - ▶ `num = s.count("s")` stores the result in the variable `num`, for later.

More on Strings: String Methods

```
s = "FridaysSaturdaysSundays"  
num = s.count("s")
```

- The first line creates a variable, called `s`, that stores the string: `"FridaysSaturdaysSundays"`
- There are many useful functions for strings (more in Lab 2).
- `s.count(x)` will count the number of times the pattern, `x`, appears in `s`.
 - ▶ `s.count("s")` counts the number of lower case `s` that occurs.
 - ▶ `num = s.count("s")` stores the result in the variable `num`, for later.
 - ▶ What would `print(s.count("sS"))` output?

More on Strings: String Methods

```
s = "FridaysSaturdaysSundays"  
num = s.count("s")
```

- The first line creates a variable, called `s`, that stores the string: "FridaysSaturdaysSundays"
- There are many useful functions for strings (more in Lab 2).
- `s.count(x)` will count the number of times the pattern, `x`, appears in `s`.
 - ▶ `s.count("s")` counts the number of lower case `s` that occurs.
 - ▶ `num = s.count("s")` stores the result in the variable `num`, for later.
 - ▶ What would `print(s.count("sS"))` output?
 - ▶ What about:

```
mess = "10 20 21 9 101 35"  
mults = mess.count("0 ")  
print(mults)
```

More on Strings...

Name:

EmpID:

CSci 127 Final, V1, F17

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[:-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:



- Don't leave it blank— write what you know & puzzle out as much as possible:

More on Strings...

Name:

EmpID:

CSci 127 Final, V1, F17

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[:-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:



- Don't leave it blank– write what you know & puzzle out as much as possible:

There are 3 fun days in a week
Two of them are ???
My favorite ??? is Saturday.

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"  
days = s[:-1].split("s")
```

- Strings are made up of individual characters (letters, numbers, etc.)

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"  
days = s[:-1].split("s")
```

- Strings are made up of individual characters (letters, numbers, etc.)
- Useful to be able to refer to pieces of a string, either an individual location or a “substring” of the string.

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"  
days = s[:-1].split("s")
```

- Strings are made up of individual characters (letters, numbers, etc.)
- Useful to be able to refer to pieces of a string, either an individual location or a “substring” of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"  
days = s[:-1].split("s")
```

- Strings are made up of individual characters (letters, numbers, etc.)
- Useful to be able to refer to pieces of a string, either an individual location or a “substring” of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"  
days = s[:-1].split("s")
```

- Strings are made up of individual characters (letters, numbers, etc.)
- Useful to be able to refer to pieces of a string, either an individual location or a “substring” of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

- `s[0]` is

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"  
days = s[:-1].split("s")
```

- Strings are made up of individual characters (letters, numbers, etc.)
- Useful to be able to refer to pieces of a string, either an individual location or a “substring” of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

- `s[0]` is 'F'.

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"  
days = s[:-1].split("s")
```

- Strings are made up of individual characters (letters, numbers, etc.)
- Useful to be able to refer to pieces of a string, either an individual location or a “substring” of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

- `s[1]` is

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"  
days = s[:-1].split("s")
```

- Strings are made up of individual characters (letters, numbers, etc.)
- Useful to be able to refer to pieces of a string, either an individual location or a “substring” of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

- `s[1]` is 'r'.

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"  
days = s[:-1].split("s")
```

- Strings are made up of individual characters (letters, numbers, etc.)
- Useful to be able to refer to pieces of a string, either an individual location or a “substring” of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

- `s[-1]` is

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"  
days = s[:-1].split("s")
```

- Strings are made up of individual characters (letters, numbers, etc.)
- Useful to be able to refer to pieces of a string, either an individual location or a “substring” of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

- `s[-1]` is 's'.

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"  
days = s[:-1].split("s")
```

- Strings are made up of individual characters (letters, numbers, etc.)
- Useful to be able to refer to pieces of a string, either an individual location or a “substring” of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

- `s[3:6]` is

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"  
days = s[:-1].split("s")
```

- Strings are made up of individual characters (letters, numbers, etc.)
- Useful to be able to refer to pieces of a string, either an individual location or a “substring” of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

- `s[3:6]` is 'day'.

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"  
days = s[:-1].split("s")
```

- Strings are made up of individual characters (letters, numbers, etc.)
- Useful to be able to refer to pieces of a string, either an individual location or a “substring” of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

- `s[:3]` is

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"  
days = s[:-1].split("s")
```

- Strings are made up of individual characters (letters, numbers, etc.)
- Useful to be able to refer to pieces of a string, either an individual location or a “substring” of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

- `s[:3]` is 'Fri'.

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"  
days = s[:-1].split("s")
```

- Strings are made up of individual characters (letters, numbers, etc.)
- Useful to be able to refer to pieces of a string, either an individual location or a “substring” of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

- `s[:-1]` is

More on Strings: Indexing & Substrings

```
s = "FridaysSaturdaysSundays"  
days = s[:-1].split("s")
```

- Strings are made up of individual characters (letters, numbers, etc.)
- Useful to be able to refer to pieces of a string, either an individual location or a “substring” of the string.

0	1	2	3	4	5	6	7	8	...	16	17	18	19	20	21	22
F	r	i	d	a	y	s	S	a	...	S	u	n	d	a	y	s
												...	-4	-3	-2	-1

- `s[:-1]` is 'FridaysSaturdaysSunday'.
(no trailing 's' at the end)

More on Strings: Splits

```
s = "FridaysSaturdaysSundays"  
days = s[:-1].split("s")
```

- `split()` divides a string into a list.

More on Strings: Splits

```
s = "FridaysSaturdaysSundays"  
days = s[:-1].split("s")
```

- `split()` divides a string into a list.
- Cross out the delimiter, and the remaining items are the list.

More on Strings: Splits

```
s = "FridaysSaturdaysSundays"  
days = s[:-1].split("s")
```

- `split()` divides a string into a list.
- Cross out the delimiter, and the remaining items are the list.

"Friday~~s~~Saturday~~s~~Sunday"

More on Strings: Splits

```
s = "FridaysSaturdaysSundays"  
days = s[:-1].split("s")
```

- `split()` divides a string into a list.
- Cross out the delimiter, and the remaining items are the list.

```
"FridaysSaturdaysSunday"  
days = ['Friday', 'Saturday', 'Sunday']
```

More on Strings: Splits

```
s = "FridaysSaturdaysSundays"  
days = s[:-1].split("s")
```

- `split()` divides a string into a list.
- Cross out the delimiter, and the remaining items are the list.

```
"FridayXSaturdayXSunday"  
days = ['Friday', 'Saturday', 'Sunday']
```

- Different delimiters give different lists:

More on Strings: Splits

```
s = "FridaysSaturdaysSundays"  
days = s[:-1].split("s")
```

- `split()` divides a string into a list.
- Cross out the delimiter, and the remaining items are the list.

```
"FridaysSaturdaysSunday"  
days = ['Friday', 'Saturday', 'Sunday']
```

- Different delimiters give different lists:

```
days = s[:-1].split("day")
```

More on Strings: Splits

```
s = "FridaysSaturdaysSundays"  
days = s[:-1].split("s")
```

- `split()` divides a string into a list.
- Cross out the delimiter, and the remaining items are the list.

```
"FridaysSaturdaysSunday"  
days = ['Friday', 'Saturday', 'Sunday']
```

- Different delimiters give different lists:

```
days = s[:-1].split("day")  
"FrixxxsSaturxxxsSundxxx"
```

More on Strings: Splits

```
s = "FridaysSaturdaysSundays"  
days = s[:-1].split("s")
```

- `split()` divides a string into a list.
- Cross out the delimiter, and the remaining items are the list.

```
"FridaysSaturdaysSunday"  
days = ['Friday', 'Saturday', 'Sunday']
```

- Different delimiters give different lists:

```
days = s[:-1].split("day")  
"FrixxxsSaturxxxsSunxxx"  
days = ['Fri', 'sSatur', 'sSun']
```

More on Strings...

Name:

EmpID:

CSci 127 Final, V1, F17

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[:-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:



- Don't leave it blank— write what you know & puzzle out as much as possible:

More on Strings...

Name:

EmpID:

CSci 127 Final, V1, F17

1. (a) What will the following Python code print:

```
s = "FridaysSaturdaysSundays"
num = s.count("s")
days = s[:-1].split("s")
print("There are", num, "fun days in a week")
mess = days[0]
print("Two of them are", mess, days[-1])
result = ""
for i in range(len(mess)):
    if i > 2:
        result = result + mess[i]
print("My favorite", result, "is Saturday.")
```

Output:



- Don't leave it blank– write what you know & puzzle out as much as possible:

There are 3 fun days in a week
Two of them are Friday Sunday
My favorite ??? is Saturday.

Lecture Slip

1. What is printed? Write your answer for each in the output box.

```
months = ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"]
#Indices:  0      1      2      3      4      5      6      7      8      9     10     11
#Or:                                     ....   -3     -2     -1
```

Output:

```
half = months[6]
print(half.upper())
```

```
print(months[-1].lower())
```

```
start = 9
print(months[start-1])
```

```
term = 3
print(months[(start+term-1)%12])
```

Recap

- On lecture slip, write down a topic you wish we had spent more time (and why).

```
1 #Predict what will be printed:
2 for i in range(4):
3     print('The world turned upside down')
4 for j in [0,1,2,3,4,5]:
5     print(j)
6 for count in range(6):
7     print(count)
8 for color in ['red', 'green', 'blue']:
9     print(color)
10 for i in range(2):
11     for j in range(2):
12         print('Look around,')
13     print('How lucky we are to be alive!')
```

Recap

- On lecture slip, write down a topic you wish we had spent more time (and why).
- In Python, we introduced:

```
1 #Predict what will be printed:
2 for i in range(4):
3     print('The world turned upside down')
4 for j in [0,1,2,3,4,5]:
5     print(j)
6 for count in range(6):
7     print(count)
8 for color in ['red', 'green', 'blue']:
9     print(color)
10 for i in range(2):
11     for j in range(2):
12         print('Look around,')
13     print('How lucky we are to be alive!')
```

Recap

- On lecture slip, write down a topic you wish we had spent more time (and why).
- In Python, we introduced:
 - ▶ For-loops

```
1 #Predict what will be printed:
2 for i in range(4):
3     print('The world turned upside down')
4 for j in [0,1,2,3,4,5]:
5     print(j)
6 for count in range(6):
7     print(count)
8 for color in ['red', 'green', 'blue']:
9     print(color)
10 for i in range(2):
11     for j in range(2):
12         print('Look around,')
13     print('How lucky we are to be alive!')
```

Recap

- On lecture slip, write down a topic you wish we had spent more time (and why).
- In Python, we introduced:
 - ▶ For-loops
 - ▶ `range()`

```
1 #Predict what will be printed:
2 for i in range(4):
3     print('The world turned upside down')
4 for j in [0,1,2,3,4,5]:
5     print(j)
6 for count in range(6):
7     print(count)
8 for color in ['red', 'green', 'blue']:
9     print(color)
10 for i in range(2):
11     for j in range(2):
12         print('Look around,')
13     print('How lucky we are to be alive!')
```

Recap

- On lecture slip, write down a topic you wish we had spent more time (and why).
- In Python, we introduced:
 - ▶ For-loops
 - ▶ `range()`
 - ▶ Variables: ints and strings

```
1 #Predict what will be printed:
2 for i in range(4):
3     print('The world turned upside down')
4 for j in [0,1,2,3,4,5]:
5     print(j)
6 for count in range(6):
7     print(count)
8 for color in ['red', 'green', 'blue']:
9     print(color)
10 for i in range(2):
11     for j in range(2):
12         print('Look around,')
13     print('How lucky we are to be alive!')
```

Recap

- On lecture slip, write down a topic you wish we had spent more time (and why).
- In Python, we introduced:
 - ▶ For-loops
 - ▶ `range()`
 - ▶ Variables: ints and strings
 - ▶ Some arithmetic

```
1 #Predict what will be printed:
2 for i in range(4):
3     print('The world turned upside down')
4 for j in [0,1,2,3,4,5]:
5     print(j)
6 for count in range(6):
7     print(count)
8 for color in ['red', 'green', 'blue']:
9     print(color)
10 for i in range(2):
11     for j in range(2):
12         print('Look around,')
13     print('How lucky we are to be alive!')
```

Recap

- On lecture slip, write down a topic you wish we had spent more time (and why).
- In Python, we introduced:
 - ▶ For-loops
 - ▶ `range()`
 - ▶ Variables: ints and strings
 - ▶ Some arithmetic
 - ▶ String concatenation

```
1 #Predict what will be printed:
2 for i in range(4):
3     print('The world turned upside down')
4 for j in [0,1,2,3,4,5]:
5     print(j)
6 for count in range(6):
7     print(count)
8 for color in ['red', 'green', 'blue']:
9     print(color)
10 for i in range(2):
11     for j in range(2):
12         print('Look around,')
13     print('How lucky we are to be alive!')
```


Recap

- On lecture slip, write down a topic you wish we had spent more time (and why).
- In Python, we introduced:
 - ▶ For-loops
 - ▶ `range()`
 - ▶ Variables: ints and strings
 - ▶ Some arithmetic
 - ▶ String concatenation
 - ▶ Functions: `ord()` and `char()`

```
1 #Predict what will be printed:
2 for i in range(4):
3     print('The world turned upside down')
4 for j in [0,1,2,3,4,5]:
5     print(j)
6 for count in range(6):
7     print(count)
8 for color in ['red', 'green', 'blue']:
9     print(color)
10 for i in range(2):
11     for j in range(2):
12         print('Look around,')
13     print('How lucky we are to be alive!')
```

Recap

- On lecture slip, write down a topic you wish we had spent more time (and why).
- In Python, we introduced:
 - ▶ For-loops
 - ▶ `range()`
 - ▶ Variables: ints and strings
 - ▶ Some arithmetic
 - ▶ String concatenation
 - ▶ Functions: `ord()` and `char()`
 - ▶ String Manipulation

```
1 #Predict what will be printed:
2 for i in range(4):
3     print('The world turned upside down')
4 for j in [0,1,2,3,4,5]:
5     print(j)
6 for count in range(6):
7     print(count)
8 for color in ['red', 'green', 'blue']:
9     print(color)
10 for i in range(2):
11     for j in range(2):
12         print('Look around,')
13     print('How lucky we are to be alive!')
```

Recap

```
1 #Predict what will be printed:
2 for i in range(4):
3     print('The world turned upside down')
4 for j in [0,1,2,3,4,5]:
5     print(j)
6 for count in range(6):
7     print(count)
8 for color in ['red', 'green', 'blue']:
9     print(color)
10 for i in range(2):
11     for j in range(2):
12         print('Look around,')
13     print('How lucky we are to be alive!')
```

- On lecture slip, write down a topic you wish we had spent more time (and why).
- In Python, we introduced:
 - ▶ For-loops
 - ▶ range()
 - ▶ Variables: ints and strings
 - ▶ Some arithmetic
 - ▶ String concatenation
 - ▶ Functions: ord() and char()
 - ▶ String Manipulation
- Pass your lecture slips to the end of the rows for the UTA's to collect.

Practice Quiz & Final Questions



- Since you must pass the final exam to pass the course, we end every lecture with final exam review.

Practice Quiz & Final Questions



- Since you must pass the final exam to pass the course, we end every lecture with final exam review.
- Pull out something to write on (not to be turned in).

Practice Quiz & Final Questions



- Since you must pass the final exam to pass the course, we end every lecture with final exam review.
- Pull out something to write on (not to be turned in).
- Lightning rounds:

Practice Quiz & Final Questions



- Since you must pass the final exam to pass the course, we end every lecture with final exam review.
- Pull out something to write on (not to be turned in).
- Lightning rounds:
 - ▶ write as much you can for 60 seconds;

Practice Quiz & Final Questions



- Since you must pass the final exam to pass the course, we end every lecture with final exam review.
- Pull out something to write on (not to be turned in).
- Lightning rounds:
 - ▶ write as much you can for 60 seconds;
 - ▶ followed by answer; and

Practice Quiz & Final Questions



- Since you must pass the final exam to pass the course, we end every lecture with final exam review.
- Pull out something to write on (not to be turned in).
- Lightning rounds:
 - ▶ write as much you can for 60 seconds;
 - ▶ followed by answer; and
 - ▶ repeat.

Practice Quiz & Final Questions



- Since you must pass the final exam to pass the course, we end every lecture with final exam review.
- Pull out something to write on (not to be turned in).
- Lightning rounds:
 - ▶ write as much you can for 60 seconds;
 - ▶ followed by answer; and
 - ▶ repeat.
- Past exams are on the webpage (under [Final Exam Information](#)).

Practice Quiz & Final Questions



- Since you must pass the final exam to pass the course, we end every lecture with final exam review.
- Pull out something to write on (not to be turned in).
- Lightning rounds:
 - ▶ write as much you can for 60 seconds;
 - ▶ followed by answer; and
 - ▶ repeat.
- Past exams are on the webpage (under [Final Exam Information](#)).
- We're starting with Spring 2018, Mock Exam.

Writing Boards



- Return writing boards as you leave...